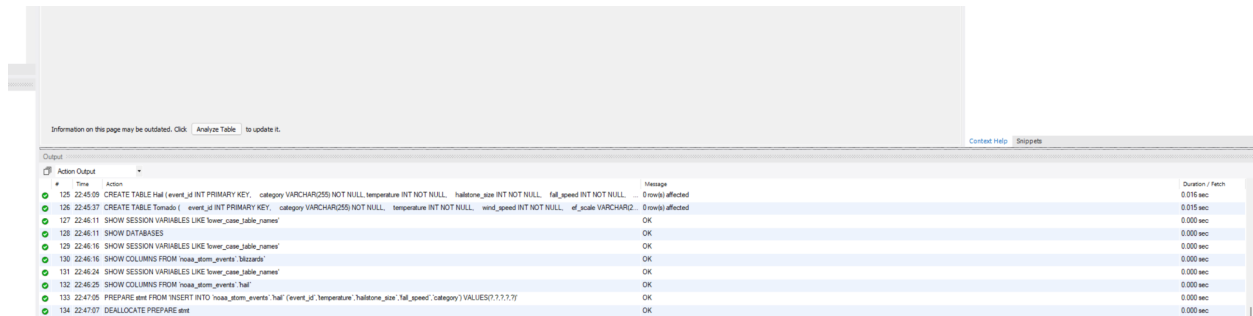


Running locally via MySQL Workbench:

Sample CSV generated with Mockaroo: <https://www.mockaroo.com/> (1000 entry limit)



The screenshot shows the MySQL Workbench interface. At the top, there's a message: "Information on this page may be outdated. Click Analyze Table to update it." Below this, the "Output" tab is active, displaying a list of SQL execution actions and their results. The actions include creating a table, showing session variables, showing databases, showing columns, and preparing a statement. The results show "OK" for most actions, indicating successful execution. The table structure for the "Hail" table is visible in the background.

#	Time	Action	Message	Duration / Fetch
125	22:45:09	CREATE TABLE Hail (event_id INT PRIMARY KEY, category VARCHAR(255) NOT NULL, temperature INT NOT NULL, hailstone_size INT NOT NULL, fall_speed INT NOT NULL, ef_scale VARCHAR(255) NOT NULL)	0 rows affected	0.015 sec
126	22:45:37	CREATE TABLE Tornado (event_id INT PRIMARY KEY, category VARCHAR(255) NOT NULL, temperature INT NOT NULL, wind_speed INT NOT NULL, ef_scale VARCHAR(255) NOT NULL)	0 rows affected	0.015 sec
127	22:46:11	SHOW SESSION VARIABLES LIKE 'lower_case_table_names'	OK	0.000 sec
128	22:46:11	SHOW DATABASES	OK	0.000 sec
129	22:46:16	SHOW SESSION VARIABLES LIKE 'lower_case_table_names'	OK	0.000 sec
130	22:46:16	SHOW COLUMNS FROM 'noaa_storm_events' 'hail'	OK	0.000 sec
131	22:46:24	SHOW SESSION VARIABLES LIKE 'lower_case_table_names'	OK	0.000 sec
132	22:46:25	SHOW COLUMNS FROM 'noaa_storm_events' 'hail'	OK	0.000 sec
133	22:47:05	PREPARE stmt FROM 'INSERT INTO 'noaa_storm_events' 'hail' (event_id, temperature, hailstone_size, fall_speed, category) VALUES(?, ?, ?, ?, ?)'	OK	0.000 sec
134	22:47:07	DEALLOCATE PREPARE stmt	OK	0.000 sec

Database Design Queries:

```
CREATE DATABASE NOAA_Storm_Events;
```

```
CREATE TABLE Category (  
    category_id INT PRIMARY KEY,  
    category_name VARCHAR(255) NOT NULL  
);
```

```
CREATE TABLE User (  
    user_id INT PRIMARY KEY,  
    user_name VARCHAR(255) NOT NULL,  
    password VARCHAR(255) NOT NULL  
);
```

```
CREATE TABLE WeatherEvent (  
    event_id INT PRIMARY KEY,  
    event_begin_time DATETIME,  
    event_end_time DATETIME,  
    damage_property INT,  
    deaths_direct INT,  
    deaths_indirect INT,  
    injuries_direct INT,  
    injuries_indirect INT,  
    damage_crops INT  
);
```

```
CREATE TABLE Tornado (  
    event_id INT PRIMARY KEY,  
    category VARCHAR(255) NOT NULL,  
    temperature INT NOT NULL,  
    wind_speed INT NOT NULL,  
    ef_scale VARCHAR(255) NOT NULL,
```

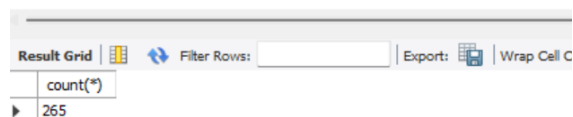
```
tornado_size INT NOT NULL,  
FOREIGN KEY (event_id) REFERENCES WeatherEvent(event_id),  
FOREIGN KEY (category) REFERENCES Category(category_name)  
);
```

```
CREATE TABLE Blizzards (  
    event_id INT PRIMARY KEY,  
    category VARCHAR(255) NOT NULL,  
    temperature INT NOT NULL,  
    wind_speed INT NOT NULL,  
    snow_depth INT NOT NULL,  
    FOREIGN KEY (event_id) REFERENCES WeatherEvent(event_id),  
    FOREIGN KEY (category) REFERENCES Category(category_name)  
);
```

```
CREATE TABLE Hail (  
    event_id INT PRIMARY KEY,  
    category VARCHAR(255) NOT NULL,  
    temperature INT NOT NULL,  
    hailstone_size INT NOT NULL,  
    fall_speed INT NOT NULL,  
    FOREIGN KEY (event_id) REFERENCES WeatherEvent(event_id),  
    FOREIGN KEY (category) REFERENCES Category(category_name)  
);
```

Sample select count (*)
FROM Categories query:

```
1 • SELECT count(*) FROM noaa_storm_events.category;
```



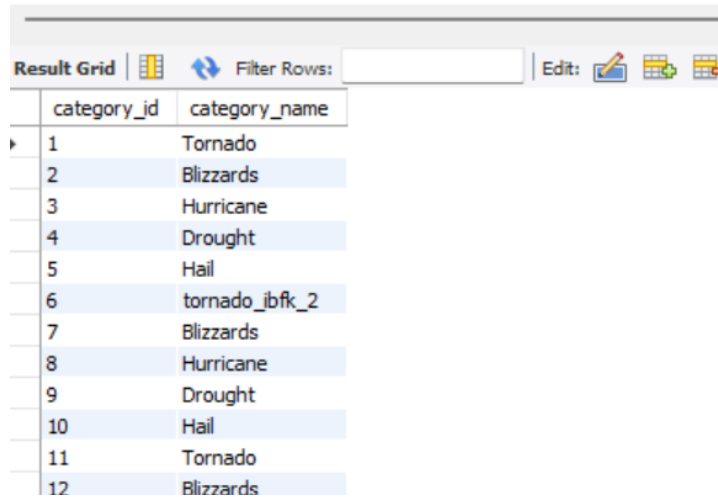
The screenshot shows a database interface with a 'Result Grid' tab. Below the tab, there is a table with one column labeled 'count(*)' and one row containing the value '265'. To the right of the table, there are options for 'Filter Rows:', 'Export:', and 'Wrap Cell C'.

count(*)
265

265 due to mockaroo CSV limit.

Sample select * from categories query:

```
1 • SELECT * FROM noaa_storm_events.category;
```



The screenshot shows a database interface with a 'Result Grid' tab. It contains a table with two columns: 'category_id' and 'category_name'. There are 12 rows of data. The first five rows have category names: Tornado, Blizzards, Hurricane, Drought, and Hail. The remaining seven rows have category names: tornado_ibfk_2, Blizzards, Hurricane, Drought, Hail, Tornado, and Blizzards. The interface also includes a 'Filter Rows' field and an 'Edit' button.

category_id	category_name
1	Tornado
2	Blizzards
3	Hurricane
4	Drought
5	Hail
6	tornado_ibfk_2
7	Blizzards
8	Hurricane
9	Drought
10	Hail
11	Tornado
12	Blizzards

Complex Queries:

Selection of the count of tornadoes directly resulting in a death toll above 20 per category of tornado.

```
SELECT count(*)  
FROM Tornado JOIN WeatherEvent on (Tornado.event_id = WeatherEvent.event_id)  
WHERE deaths_direct > 20  
GROUP BY category;
```

Selection of the size of hailstone typical of a hail storm resulting in between \$2000 and \$4000 of property damage.

```
SELECT Hail.hailstone_size  
FROM Hail  
WHERE NOT IN (  
    SELECT hailstone_size  
    FROM Hail JOIN WeatherEvent ON Hail.event_id = WeatherEvent.event_id  
    WHERE WeatherEvent.damage_property < 2000  
  
    UNION
```

```
SELECT hailstone_size
FROM Hail JOIN WeatherEvent ON Hail.event_id = WeatherEvent.event_id
WHERE WeatherEvent.damage_property > 4000
);
```